ESTIMATING SOFTWARE PROJECT REQUIREMENTS FOR RESOLVING DEFECT BACKLOGS

FIELD OF THE INVENTION

{001} The present invention concerns the field of software development, and more particularly concerns estimating software development project requirements for resolving dynamically generated defect backlogs.

BACKGROUND

{002} An important aspect of the software development process centers on understanding the resources needed to complete projects on-time and according to budget. This often involves real-time forecasting of the completion dates for dynamically generated populations of tasks.

{003} Such tasks may arise as a product of the various test phases of the development process, such as functional verification. Here, pre-planned units of work such as test cases lead to the discovery of defects, which must be fixed. Hence, further units of work are generated dynamically, as defects are discovered during execution of the test cases.

{004} At present, project planning methods for dealing with the dynamically generated work items are imprecise at best. For example, a predetermined percentage of the development team budget may be forecast to cover fixing the defects discovered during the functional verification tests.

{005} Unfortunately, such methods suffer from important drawbacks. Principal among these is the challenge presented to day-to-day project management, which must decide whether the current allocation of resources is adequate to resolve the current backlog of defects in the allotted

time. The primary methods for deciding this question are generally ad hoc, as the typical defect lifetime is shorter than the minimum planning unit for most projects; moreover, the cost of updating the project plan with the newly identified tasks would be prohibitive.

{006} Often, the ad hoc forecasts result in overly optimistic estimates of the available capacity of the development team. As a result, development resources may be over-committed to draining the defect backlog, at which point fewer resources are available for the development of new code. In the worst case, this may lead to a positive feedback situation, wherein the increasingly thinly stretched developers generate increasingly more defects, which necessitate the shift of an ever-larger portion of the available resources from developing new code to draining the defect backlog.

{007} Thus there is a need for a better way to forecast the completion date for a dynamically generated defect backlog, or, conversely, to forecast the resources needed to drain a dynamically generated defect backlog by a specified date.

SUMMARY

{008} Aspects of the invention include methods, apparatus, and computer program products for analyzing defect backlogs that arise in the software development process. Analysis is based on a validity ratio that projects the number of open defects that are likely to actually require fixes, a fix rate that describes the performance of the team charged with fixing the defects, defect census data, and team performance census data. One outcome of the analysis may be an estimate of the date by which the defect backlog is expected to be resolved. Another outcome of the analysis may be an estimate of the capacity of a team to resolve defects between a given start date and a given target date.

BRIEF DESCRIPTION OF THE DRAWINGS

{009} FIG. 1 is a flowchart showing computation of a fix rate and a validity ratio.

{010} FIG. 2 is a flowchart that shows an exemplary process for estimating a drain date for a defect backlog, using the fix rate and the validity ratio.

{011} FIG. 3 is a flowchart that shows an exemplary process for estimating the capacity of a development team to fix defects between given start and target dates, using the fix rate and the validity ratio.

{012} FIG. 4 is a block diagram that shows exemplary apparatus suitable for executing the processes of FIGs. 1-3.

DETAILED DESCRIPTION

{013} Software defects may be modeled using six states. The states, which are described below, are: open, working, verify, closed, returned, and canceled.

{014} Tracking through the states of the model begins when a defect is discovered and a bug report is filed. At this point, the newly found defect is in the open state. A developer then evaluates the defect, and determines whether a fix is required. If so, the defect moves to the working state, and the developer opens an associated track, codes a fix for the defect, and gives the track "integrate status." The code changes in tracks having integrate status are applied to the source code when a new software build is started.

{015} The build script is then executed. If the execution completes successfully, the defect moves to the verify state. The originator of the defect, or the originator's representative, then

examines the execution of the altered source code for correctness. If the originator or the representative concludes that the defect has been fixed satisfactorily, the defect moves to the closed state, which indicates that the defect needs no further attention.

{016} When the developer who evaluates the defect determines that a fix is not required, the defect moves to the returned state. A defect in the returned state may be re-opened, perhaps with additional information, or may move to the canceled state, at the option of the defect report originator. The model's canceled state and closed state are equivalent, in the sense that a defect in either of these states requires no further work.

{017} Defect census data for each date of relevant activity is kept in a defect census data repository, which may reside in, for example, a spreadsheet or a database. The defect census data summarizes the numbers of defects in each of the states defined by the six-state model. The persistence of the defect census data preferably covers at least five software builds.

{018} Defects are fixed by teams of developers or by individual developers, both of which are called here "teams" for descriptive convenience. For each team, a fix rate may be computed from the defect census data, as shown in FIG. 1. The number of defects in the closed state and the number of defects in the verify state are added together (step 100), and the resulting sum is divided by the number of working days for the team, thereby giving the fix rate (step 110). The number of working days is the number of days the team spent actually working on resolving the defects in the defect census data repository, as opposed to working on other development activities, or holidays, weekends, and so forth. For example, at the end of five builds taking a total of 20 working days, a team may have sent 100 defects to the closed state and have 20 defects remaining in the verify state. The fix rate for that team would be 120 divided by 20, or six per working day.

{019} Over the course of time, statistics may be collected regarding fix rates, giving average and

-4-

Nth-percentile fix rates for the various teams. Here, these are called team performance census data. The team performance census data may be stored in a team performance census data repository, which may be held in, for example, a spreadsheet or database.

{020} The proportion of the defects in the open state that are expected to actually require a fix is described by a validity ratio. The validity ratio may be computed as further shown in FIG. 1. The numbers of defects in the working, verify, and closed states are added together, to provide a sum B (step 120). The numbers of defects in the canceled, working, verify, and closed states are added together to provide a sum C (step 130). B is divided by C to provide the validity ratio (step 140). Again, counts of the numbers of defects in the various states are taken over a five-build span of time. For example, suppose that the number of defects in the working state is six, the number of defects in the verify state is six, the number of defects in the closed state is 100, and the number of defects in the canceled state is eight. In this case, the validity ratio is about 0.933.

{021} The defect backlog for a given project or build may be analyzed using the fix rate and the validity ratio. One outcome of the analysis may be a forecast of when the defect backlog should be resolved or drained, i.e., the date by which all of the defects are expected to be in the closed or canceled states. This date is called here the "drain date."

{022} An exemplary process for estimating the drain date is shown in FIG. 2. To compute the estimate, the number of defects in the open state is multiplied by the validity ratio, and the resulting product is added to the number of defects in the working state (step 200). This sum may be called the "work_left." The work left is then divided by the fix rate (step 210). This quotient may be called the "days_ left." The quotient days_left is then converted to the drain date with respect to the starting date, by adjusting for non-working days (step 220).

{023} For example, suppose that there are ninety-one defects in the open state, and eleven

defects in the working state. Continuing the example begun above, wherein the validity ratio is about 0.933, gives a work_left value of about 96 defects. Using a fix rate of 6 defects per day gives about 16 for the value of days_left. Suppose, for example, that the start date, which coincides with the date that the drain date is estimated, is May 5, 2004. Suppose that Saturdays and Sundays are not working days, that May 31 is a holiday, and that the team in question is scheduled for other duties on May 12, 13, and 14, which therefore do not count as working days. The drain date will then be about 16 working days from May 5, which is June 1, 2004.

{024} Above, an average fix rate was used to compute days_left, and thus the drain date. As mentioned earlier, N-th percentile team performance census data may be kept. This data may be used to generate N-th percentile estimates of the drain date. Intuitively, the more frequently a team has produced at a rate exceeding a given fix rate, the more confidence can be placed in the assumption that the team will produce at a rate exceeding the given fix rate in the future. Thus, observed fix rates at the 99th, 90th, 75th, 50th, 25th, and 10th percentiles may be kept in the team performance census data repository, and used to provide corresponding drain dates having confidence levels from 1-90%. In another embodiment, the various fix-rate percentile estimates may be derived from the moments of a probability density function that is believed to fit the work rate of a particular team, or teams in general, based on empirical or theoretical considerations.

{025} Another outcome of the analysis of the defect backlog may be a forecast of the defect processing capacity remaining between a start date and a predetermined target date. An exemplary process for estimating this capacity is shown in FIG. 3. The number of working days between the start date and the target data is determined (step 300). The number of working days is then multiplied by the fix rate (step 310), and the resulting product is divided by the validity ratio (step 320). The resulting quotient is the remaining capacity, stated in numbers of defects. For example, suppose that there are ten working days between the start date and the target date. If the fix rate is six per day, and the validity ration 0.933, the capacity of the team over the ten

working days is about 64 defects. Again, the N-th percentile statistics of the fix rate may be used to compute corresponding confidence levels of the capacity.

{026} FIG. 4 shows exemplary structure of apparatus suitable for use according to the present invention. A defect census data repository 400 holds the defect census data, a team performance census data repository 410 holds team performance census data, and an estimation engine 420 performs the computations mentioned above in the discussions concerning FIGs. 1-3. This particular structure is shown only for the sake of descriptive clarity with regard to these earlier discussions, and is not limiting of the invention. In practice, the functions of the blocks of FIG. 4 may be performed by, for example, the various elements of a personal computer, workstation, server, and the like. The repositories and the estimation engine may be embodied partly or fully by a spreadsheet or other mathematical software executed by a programmable processor such as a processor in a personal computer, workstation, server, and the like.

{027} The present invention also encompasses computer program products, including program storage devices readable by a machine, tangibly embodying programs of instructions executable by the machine for implementing the methods and apparatus described above. The program storage device may take the form of any media that can contain, store, communicate, propagate, or transport the program for use by the machine. These media include, for example, computer diskettes, RAM, ROM, CD, EPROM, communication media for transferring instructions, and the like.

{028} Although the foregoing has described methods, apparatus, and computer program products for estimating the drain date of a software defect backlog and for estimating the capacity of a team to fix defects over a specified period of time, the description is illustrative of the invention rather than limiting, and the invention is limited only by the appended claims.